A Probabilistic Constructive Approach To Optimization Problems

Jennifer L. Wong, Farinaz Koushanfar†, Seapahn Meguerdichian, Miodrag Potkonjak
Computer Science Department, University of California, Los Angeles, CA 90095
†EECS Department, University of California, Berkeley, CA 94720

ABSTRACT

We propose a new optimization paradigm for solving intractable combinatorial problems. The technique, named Probabilistic Constructive (PC), combines the advantages of both constructive and probabilistic algorithms. The constructive aspect provides relatively short runtime and makes the technique amenable for the inclusion of insights through heuristic rules. The probabilistic nature facilitates a flexible trade-off between runtime and the quality of solution.

In addition to presenting the generic technique, we apply it to the Maximal Independent Set problem. Extensive experimentation indicates that the new approach provides very attractive trade-offs between the quality of the solution and runtime, often outperforming the best previously published approaches.

1. Introduction

In order to build high quality CAD software, a number of components need to be in place. These components include proper abstractions of synthesis problems that capture important features and eliminate non-important ones, and models that characterize design components such as delay, area, and early power prediction. Any developed software must be modular and written in such a way that it can be easily reused and modified. Furthermore, there is a strong demand for user interfaces that simplify the designer's interaction with CAD tools during the design process. While the list of desired CAD software components is long, at the heart of all synthesis software are optimization algorithms for solving computationally intractable problems.

It is interesting and enlightening to classify the developed algorithms. *Figure 1* shows the classification according to two main criteria: (*i*) the way in which the solution is built and (*ii*) the presence or absence of randomness. More specifically, all algorithms can be classified as either *deterministic* or *probabilistic* in one dimension, and as *constructive* or *iterative improvement* in the second dimension. The largest group of algorithms are *deterministic constructive*. For example, many CAD algorithms are based on the forced directed paradigm or use dynamic programming. In the last three decades, deterministic iterative improvement algorithms [15] were proposed for many problems and were able to produce the best results. In particular, deterministic iterative improvement algorithms are widely and frequently used for partitioning [2]. Since the mid-80's, when *Simulated Annealing* was first proposed for use in designing multi-chip computers [16], probabilistic iterative improvement has attracted a great deal of attention for solving CAD problems. Techniques such as *Genetic Algorithms*, *Tabu Search*, and *Simulated Evolution*, due to their programming simplicity and flexibility, have been used for a variety of synthesis tasks. Their main disadvantage however, is usually long runtime.

While numerous algorithms populate three of the quadrants in *Figure 1*, the probabilistic constructive quadrant is empty. The closest in spirit to this quadrant are randomized deterministic algorithms [19]. Our goal in this paper is to push the envelope well beyond this type of randomization and develop algorithms that are simultaneously constructive and probabilistic, by leveraging on the positive properties of both constructive algorithms and probabilistic algorithms. The main advantage of constructive algorithms is* their relatively short runtime and flexibility to incorporate a variety of insights as efficient heuristics. On the other hand, the main advantage of probabilistic algorithms is their inherent flexibility that facilitates the trade-off between quality of solution and runtime.

The new approach can best be explained at the intuitive level in the following way: We start by searching for a small part of the solution that can be solved effectively, in such a way that the remainder of the problem is also suitable for similar optimization. For this search, we propose a probabilistic methodology, where parts of the solution are considered, and the decision of which to select is made in a probabilistic manner so that the likelihood of obtaining a high quality solution is maximized. The quality of the solution is evaluated using an objective function. After the small part is solved, we eliminate it from further consideration and solve the remaining problem iteratively using the same approach. The final stage is to incorporate the solutions to each of the small parts together to form the final solution to the problem.

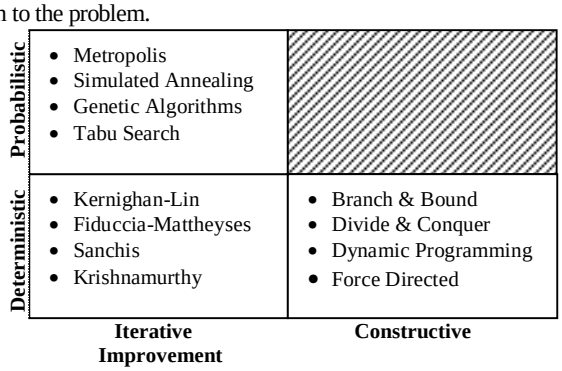

Figure 1 - Classification of Optimization Algorithms

2. Related Work

By far the most popular and widely used generic algorithmic paradigm is the deterministic constructive approach. Algorithms of this type have been applied on a vast variety of problems, starting from sorting and basic graph algorithms such as *Breadth First Search* and *Topological Sort*, to more complex graph algorithms, such as *All-Pairs Shortest Path* and *Maximum Flow*. Several generic algorithmic techniques of the constructive deterministic approach have found many applications. For example, *Greedy Algorithms*, *Dynamic Programming*, and *Branch-and-Bound* are used to solve many different problems.

In 1970, Kernighan and Lin introduced the first iterative improvement heuristics, which was applied for graph partitioning [15]. The algorithm uses pair swap moves to iteratively reassign elements to different partitions. It proceeds in a series of passes, during which each component is moved exactly once. A number of improvements on the basic strategies have been

This work partially supported by the National Science Foundation (NSF) under Grant No. NI-0085773. Any opinions, findings and conclusions expressed in this material are those of the author(s) and do not necessarily reflect the views of the NSF.

proposed over the years [10]. An excellent survey of this research is given in [2]. The iterative improvement paradigm has been applied to many other optimization problems, including the Traveling Salesman Problem [17].
improves the average runtime of algorithms. There are two types of randomized algorithms, *Las Vegas* and *Monte Carlo*. *Las Vegas* algorithms always generate a correct solution, but their runtimes vary depending on the distribution of inputs. In contrast, *Mote Carlo* algorithms may sometimes produce an incorrect solution, but they run in a predictable amount of time. The probability of a *Monte Carlo* algorithm producing an incorrect solution can be made arbitrarily small by repetitively running the algorithm, each with independent random choices.

Since 1953, a number of probabilistic iterative improvement algorithms have been proposed. Two of them have origins in statistical mechanics: the *Metropolis* algorithm [19] and *Simulated Annealing* [16]. *Simulated Annealing* found a spectrum of application in engineering, computer science and image recognition [1]. In contrast to deterministic iterative improvement algorithms, *Simulated Annealing* allows hill-climbing moves. Consequently, a number of probabilistic iterative improvement algorithms that often explore with analogy to physical and biological world have been proposed including *Genetic Algorithms* [13,20], *Neural Networks* [14], *Simulated Evolution* [7,12], and *Tabu Search* [8,11].

The new probabilistic constructive paradigm is different from all the above paradigms. In some sense it is closest to randomized algorithms. Conceptually, the difference is that probabilistic constructive uses extensive probabilistic search to find an attractive way to solve an arbitrary small part of the problem and construct (as opposite to improve) the solution.

3. Generic Probabilistic Constructive Approach

The basic idea behind the probabilistic constructive approach is to search, probabilistically, for a small part of the solution which can be solved well and which leaves the remaining problem amenable for further optimization. For example, when we are searching for a graph coloring solution, we can color a few nodes in a particular way and remove them from further consideration.

The generic approach has the following ten components:

Candidate Part (CP): The candidate part is a relatively small portion of the problem that can be efficiently solved in a particular way. In the general case, we must make two choices regarding the *CP*: (*i*) which components of the problem to consider and (*ii*) how to resolve that part of the problem. It is important that the *CP* is not too small in order to avoid overly local and greedy solutions. It is also important that the *CP* is not too large in order to avoid long search times. For example, in Graph Coloring, coloring a single node at a time is a *CP* decision that is too local. However, it is difficult to find a promising coloring solution if we decide to color too many nodes simultaneously.

Probabilistic Search (PS): One of the more important aspects of the algorithm is how to efficiently search the solution space using probabilistic constructs. There are two main alternatives. One is to define a move that probabilistically replaces a single component from the *CP* with the new component. The second method is to generate a new *CP* from scratch. The first technique is faster while the second is capable of quickly scanning the complete solution space. From the implementation point of view, random number generation is a computationally intensive task in the probabilistic constructive algorithm. In our implementation, we use a stored list of randomly generated numbers that is traversed starting from randomly selected points. While this approach generates numbers that are not completely compliant with the standard test for randomness, the extensive implementation implies that it can speed up the performance of the algorithm by an order of magnitude without sacrificing the quality of solution.

Candidate List (CL): The candidate list contains the *k* best solutions for the *CP*s found using probabilistic search. The most important criteria related to the *CL*s are the ones that select which solutions should be included in the list. The simplest approach is to include only the *k* best solutions (with *k* best

Randomized versions of the deterministic constructive algorithms have been popular for a long time [19]. Randomization often dramatically

*OF*s). A more sophisticated approach takes into account the overlap between the new proposed solution and solutions in the candidate list.

Objective Function (OF): The objective function is a heuristic measure of likelihood that a particular solution to a particular part of the final solution is a promising choice. The main trade-off here is between accuracy (ability to estimate) and runtime.

Comprehensive Objective Function (COF): The objective function is calculated for all proposed solutions and therefore it is important that the *OF* is fast. Once the number of candidates is reduced to only a few, it is essential to evaluate them as accurately as possible. Therefore, before the final selection of a particular candidate from the *CL*, we calculate the *COF*. The main difference between *OF* and *COF* is that the former involves calculations of properties related only to properties of a small part of the solution, while the latter takes into account properties of the still remaining unsolved regions. Another important criterion that needs to be taken into consideration is the overlap between the selected *CP* and other candidates from *CL*. Clearly, less overlap implies that more of the current candidates can be reused in the next stages of the algorithm.

Stopping Criteria: The effectiveness of probabilistic search for a promising *CP* is positively correlated with the search time. Nevertheless, two general guidance criteria can be stated: (*i*) longer search time is required in the beginning when the problem is still large, (*ii*) the best indication of finding a new quality solution for a *CP* is that for a long period of time no new *CP* is observed.

Best Candidate Selection: The best candidate selection is the process of selecting the part of the solution that will be accepted. The simplest strategy is to select the one with the best *COF*. One can envision a multitude of alternatives where information from the previous runs of the algorithm is considered or delayed decision is used.

Solution Integration: Divide and conquer is a popular algorithmic paradigm. Its application is often restricted due to the difficulty of integrating components. Therefore, one of the most important aspects of the probabilistic constructive approach is to develop mechanisms for integrating solutions to the small parts into the solution of the overall problem. In a sense, this is the most demanding aspect of the PC approach, which requires the highest degree of creativity. Nevertheless, there exists a generic technique for this task. The technique is based on constraint manipulation, where the already solved parts, are presented as constraints to the remaining problem. A small example will better explain this paradigm. Consider the graph-coloring problem. If we decide to color two nodes n_1 and n_2 with the same color as a *CP*, all that is needed is to replace these n_1 and n_2 with a single node n' in the remainder of the problem. Note that n' should have edges to all the nodes that were connected to n_1 and n_2.

Overall Control Strategy: Since the new approach is probabilistic, each run of the algorithm, in principle, produces different solutions and has different runtimes. One can super-impose a variety of control strategies using the generic algorithm as the building block. For example, one can use multi-starts or keep statistics about the difficulty of resolving some parts of the solution and use this as the decision criteria of when to terminate an unpromising start.

The new problem-solving paradigm can be explained in the following way: we attempt to find a small and readily solvable part of an overall problem and find a high quality solution to that part. The objective function is used to evaluate the quality of the proposed solution. Examining all parts of the problem is a procedure with exponential time complexity and therefore is not a plausible approach. This suggests the use of a randomized search algorithm. The search should avoid visiting the same parts of the problem more than once. The parts with a high solution quality are stored for future considerations. In particular, diverse solutions are very beneficial because they can be used consequently to form other parts of the solution. Furthermore, if possible, the *CP* should be flexible in order to allow the imposing of additional control or search strategies later on. The pseudo code

of a generic approach for the probabilistic constructive procedure (*GPC*) is listed in *Figure 2*.

First, the algorithm builds a *CL* of promising solvable *CP*s (CP_i). The promising candidate is found after applying the probabilistic search to the current instance of the problem, *P*. During this probabilistic selection, the algorithm favors *CP*s that are more likely to be solved efficiently (have higher *OF* values), and adds only the best *CP*s to the *CL*. Next, the comprehensive objective function, *COF*, is calculated for each of the elements in *CL*. The *BCS* is selected from the *CL* according to the corresponding rule. This selected *BCS*, or CP_i which evaluated best according to the *BCS* rules, is then integrated as part of the solution and eliminated from the problem. The procedure then repeats on the remainder of the problem until a complete solution is found.

```
while ( Overall Controll Strategy is not satisfied)
   Procedure GPC(P)
   S=∅;
   while (S(P) is not complete)
        while (stopping criteria is not satisfied)
           CP_i = GenerateCP();          //using PS
           OF_i = CalculateOF(CP_i);
           if(OF_i > OF_min)       //OF_min is CP with smallest OF in CL
              UpdateCL(CP_i);
        for(all CP_j in CL)
           CalculateCOF(CP_j);
        BCS = BestCandidateSelection(CL);
        S(P) = SolutionIntegration(S(P),BCS);
```

Figure 2 - Generic Probabilistic Constructive Algorithm

4. Application To Maximum Independent Set

We first explain how the new PC paradigm can be applied to the Maximum Independent Set problem.

Problem*: Maximum Independent Set*
Instance: *Graph G=(V,E), positive integer K ≤ |V|.*
Question: *Does G contain an independent set V', with cardinality greater than or equal to the cardinality of all other independent sets of G, i.e. a subset V'⊆V such that for all pairs of vertices u,v∈V' the edge {u,v}∉E.*

Given an undirected graph *G=(V,E)* where *V* is the set of vertices and *E* is the set of edges in *G*, an independent set is defined as a subset *V'⊆V* such that for all pairs of vertices *u,v∈V'* the edge *{u,v}∉* E. An independent set *V'* is a Maximal Independent Set, if ∀ *v∈V*, either *v∈V'* or there is a *u∈V'* such that the edge *{u,v}∈E*. An independent set *V'* is a maximum independent set, if |*V'*| is greater than or equal to the cardinality of all other independent sets of *G*. It is widely known that the independent set problem is directly related to several other key graph theory problems such as *Vertex Cover* and *Clique* [9].

The probabilistic constructive algorithm can be applied to the Maximum Independent Set problem in at least two different ways. The first is to select nodes to include in the MIS. The other way is to select nodes that are to be excluded from the MIS. In this case, the solution is the nodes that remain unconnected in the final graph.

For the first approach, selecting nodes to include in the MIS, we define the components in the following way.

Candidate Part (CP): We select any subset of nodes where there are no edges between them to be considered as the *CP*. Each *CP* is a possible subset of the nodes in the final solution, or MIS. The candidate part can be of size *k*, where *k* is a variable or constant value. In our experimental evaluations, we used *k* = 4 nodes. There are several good heuristics for selecting *k*. For example, *k* can be a fraction of the number of nodes in the graph.

Probabilistic Search: We search the solution space by excluding one node from a *CP* of size *k*, and including another node. The nodes to exclude, N_e, and include, N_i, in the *CP* are chosen according to the following equations calculated for each node:

$$N_e = w_1 n + w_2 n_u + w_3 n_1 \text{ where } n_1 = \sum_{i=1}^{\#\_neib} n(i)$$

$$N_i = \frac{1}{N_e}$$

We define *n* as the number of neighbors of the node, and n_u as the number of unique neighbors, i.e. neighbors that no other node in the *CP* have edges to. The variable n_1 is the total number of neighbors for all the neighbors of the current node. We select probabilistically which node to exclude or include according to the nodes N_e or N_i value.

Candidate List (CL): We include k_1 *CP*s in the *CL* with the constraint that no node exists in more than 1/5 of the *CP*s in the *CL*. We also state that if the *OF*s of the *CP*s are relatively consistent in value, then we continue to add *CP*s to the *CL* to make it twice as long as usual. On the other hand, if the values of the *OF* are distributed, then we cease building the list, assuming that we have satisfied the minimum list size, k_{min}. We reason that if the values of the *OF* are relatively consistent, then most likely we should continue to search further to find a good overall selection. However, if the values are wide spread, the *CL* has a good representation of the solution space.

Objective Function (OF): The objective function is the weighted sum of n_r, the number of nodes in the remainder of the graph that are still eligible to be included in the MIS, and *e* is the total number of edges minus the incident edges. We give preference to the *CP*s that leave a large number of nodes eligible for selection in the next iteration. We also give preference to the *CP*s that eliminate many edges for the graph. The less edges in the graph the more likely we are to be able to select more nodes to eventually include in the MIS.

$$OF(CP_i) = \alpha_1 n_r + \alpha_2 e$$

Comprehensive Objective Function (COF): For the *COF* we combine the *OF* with an additional component. This component penalizes a *CP* for having a large number of neighbors outside the *CP*. We denote the number of neighbors of node *i* in the *CP* by n_i. We denote the size of the *CP* by *k*.

$$COF(CP_i) = OF(mIS_i) + \alpha_3 \sum_{i=1}^{k} n_i^2$$

We penalize *CP*s with a higher number of neighbors outside of the *CP* because they limit the number of possible nodes for the next iteration. Note that in this case α_3 is negative.

Stopping Criteria: We stop searching for new *CP*s for the *CL*s after kn_r attempts to find a *CP* with an improved *OF*, where n_r is the number of remaining nodes in the graph. The idea is that if the recent searching efforts do not provide any improvement then most likely none will be found. We found that $k = 5$ performs well in practice.

Best Candidate Selection: We select the best *CP* by enhancing the *COF* with additional criteria - the number of occurrences of the *CP* nodes in the *CL*. If the nodes in the *BCS* only appear in one *CP* in the *CL*, then by selecting the *CP* we preserve a large number of already found *CP*s in the *CL* and leave a large part of the solution space with high potential untouched.

We denote the total number of appearances for node *i* in the CP as a_i.

$$BCS(CP_i) = w_1 COF(CP_i) + \frac{1}{a_i}$$

Solution Integration: We integrate the *BCS* into the solution and leave the remaining problem to be solved by removing all nodes in the selected *CP*, as well as neighbors of the nodes and all incident edges.

Overall Control Strategy: For the overall control strategy we conduct *n*/10 multi-starts given that *n* is the number of nodes in the original instance. This number was determined experimentally.

The second approach, where we select nodes to exclude from the MIS, uses many of the same component definitions as the first approach. The definitions of the *CP*, *CL*, *COF*, *Stopping Criteria*, *BCS*, *Solution*

Integration and *Overall Control Strategy* all stay the same. We define the remaining components in the following way.

Probabilistic Search: We select any one of the nodes to be excluded from the *CP* and replaced with another node. The nodes to be included and excluded are selected probabilistically using the following values:

$$N_e = \frac{1}{N_i}, N_i = \sum_{j=1}^{\#\_neib} \frac{1}{|n_{ijk}|}$$

We define the neighbor of node n_i as n_{ij}, and the neighbor of n_{ij} as n_{ijk}. Therefore we eliminate the nodes with many 2nd neighbors, because these neighbors will greatly harm a potential solution by eliminating a significant number of nodes from consideration.

Objective Function (OF): In this case we simplify the objective function to only include the number of edges which remain in the resulting graph, *e*.

$$OF(CP_i) = \alpha e$$

5. Experimental Results

In this section we present the experimental results conducted on several benchmarks. We applied the PC technique and compared the results to previously published results [5]. All testing was done on a 300-MHz Sun Ultra-10 Workstation.

Name	*V*	*E in Clique*	*E in MIS*	γ	*CPU*
school1_nsh	358	16710	47193	14	0.22
Keller4	171	9435	5100	11	0.9
sanr200_0.7	200	13868	6032	18	3.71
brock200_1	200	14834	5066	21	22.58
san200_0.7_2	200	13930	5970	18	0.31
P_hat300-2	300	21928	22922	25	0.94
Hamming8-4	256	20864	11776	16	0.006
san200_0.9_1	200	17910	1990	70	1.02
MANN_a27	378	70551	702	126	12.3

Table 1 - Experimental Results for Independent Sets

We ran testing on instances for the problem of finding the maximum clique. The maximum clique problem can be easily mapped to MIS by complementing the graph. Complemented graph G_c of graph *G* is a graph that has the same set of vertices as *G*. However, G_c has edges between two vertices if and only if *G* does not have edge between these two vertices. The *MIS* in a graph is the maximum clique in the complemented graph and vise versa.

The first column of *Table 1* indicates the name of the maximum clique instance (the instances are from [4,6]) while the second column states the number of vertices in the graph. The next two columns give the number of edges in the original graph and the number of edges in the complemented graph respectively. The fifth column represents the number of nodes in the MIS or maximum clique. In all cases, the probabilistic constructive approach was able to find the optimal solution. Finally, the sixth column displays the runtime for finding the MIS using the PC heuristic. The reported times are faster than any other previously published time [5].

6. Conclusion

We introduced a new *probabilistic constructive* algorithm paradigm. The method combines the relatively short runtime of constructive algorithms and the flexibility of probabilistic algorithms. We discussed the main components of the new approach. We applied the algorithm to the problem of Maximum Independent Set. In [21], the PC approach is applied to graph coloring and two design problems (code covering and scheduling). Extensive experimentation indicates that the new algorithm is capable of achieving competitive or better results than previously published approaches, often with shorter runtimes.

7. References

[1] E. Aarts, J. Korst. "Simulated Annealing and Boltzmann Machines: a stochastic approach to combinatorial optimization and neural computing." New York: Wiley, 1989.

[2] C.J. Alpert, A.B. Kahng. "Recent Directions in Netlist Partitioning: A Survey." Integration: The VLSI Journal, vol. 19, 1995.

[3] V. Cerny. "Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm." Journal of Optimization Theory and Applications, vol.45, pp.41-51, 1985.

[4] J. Cong, M. Hossain, N.A. Sherwani. "A provably good multilayer topological planar routing algorithm in IC layout designs." IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol.12(1), pp.70-78, 1993.

[5] O. Coudert. "Exact coloring of real-life graphs is easy." IEEE/ACM Design Automation Conference, pp.120-126, 1997.

[6] ftp://dimacs.rutgers.edu/pub/challenge/graph/benchmark/. Benchmark for graph coloring and clique, 1993.

[7] L.J. Fogel, A.J. Owens, M.J. Walsh. " Artificial Intelligence through Simulated Evolution." New York: John Wiley, 1966.

[8] C. Friden, A. Hertz, D. De Werra. "TABARIS: an exact algorithm based on Tabu Search for finding a maximum independent set in a graph," Computers & Operations Research, vol. 17(5), pp.437-45, 1990.

[9] M.R. Garey, D.S. Johnson. "Computers and Intractability: A Guide to the Theory of NP-Completeness." New York: Freeman, 1979.

[10] F. Gavril. "Algorithms for minimum coloring, maximum clique, minimum covering by cliques, and maximum independent set of a chordal graph." SIAM Journal on Computing, vol.1, pp.180-187, 1972.

[11] F. Glover. "Tabu search part I." ORSA Journal on Computing, vol.1 (3), pp.190-206, 1989.

[12] W. D. Hillis. "Co-evolving parasites improves simulated evolution as an optimization technique." Artificial Life II, pp. 313-384, 1991.

[13] J. Holland. "Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence." Ann Arbor: University of Michigan Press, 1975.

[14] J.J. Hopfield. "Neural Networks and Physical Systems with Convergent Collective Computational Properties." Proceedings of National Academy of Science (U.S.), vol.79, pp.2554-2558, 1982.

[15] B. W. Kernighan, S. Lin. "An Efficient Heuristic Procedure for Partitioning Graphs." Bell System Technical Journal, vol. 49(2), pp. 291-307, 1970.

[16] S. Kirkpatrick, C. Gelatt, M. Vecchi. "Optimization by Simulated Annealing." Science (220), pp.671-680, 1983.

[17] S. Lin, B.W. Kernighan. "An effective heuristic algorithm for the traveling-salesman problem." Operations Research, vol. 21 (2), pp. 498-516, 1973.

[18] N. Metropolis, A. Rosenbluth, R. Rosenbluth, A. Teller, & E. Teller. "Equation of state calculations by fast computing machines." Journal Chemical Physics, vol.21, pp.1087-1092, 1953.

[19] R. Motwani, P. Raghavan. "Randomized algorithms." New York: Cambridge University Press, 1995.

[20] A. Sakamoto, X. Liu, T. Shimamoto. "A Genetic Approach for Maximum Independent Set Problems." IEICE Transaction on Fundamentals, vol. E80-A (3), pp.551-556, 1997.

[21] J.L. Wong, F. Koushanfar, S. Meguerdichian, M. Potkonjak. "A Probabilistic Constructive Approach to Optimization Problems." UCLA Computer Science Department Technical Reports #010029, August 2001.